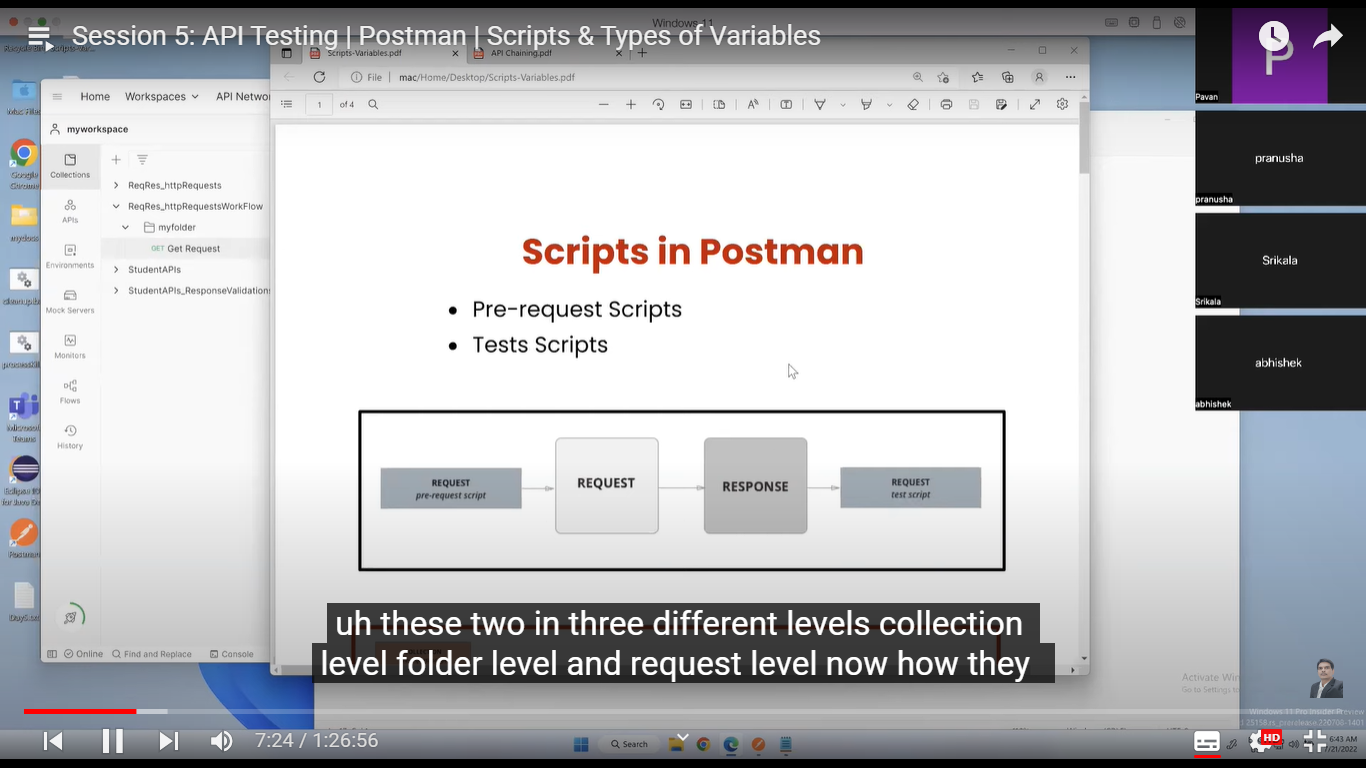
API Session – Chapter 7

- Postman Scripts -

# Script Types



# Script Levels

**Postman** offers flexibility in writing scripts at three distinct levels:

## 1. Collection Level

* **Scope:** Applies to all requests within the collection.
* **Use Case:** Define global variables, shared functions, or setup actions that are common across all requests in the collection.

## 2. Folder Level

* **Scope:** Applies to all requests within a specific folder.
* **Use Case:** Create scripts that are relevant to a group of related requests. For example, setting up environment variables specific to a particular API endpoint or service.

## 3. Request Level

* **Scope:** Applies only to the individual request.
* **Use Case:** Write scripts that are specific to the behavior of a single request, such as pre-request scripts to set dynamic parameters or post-request scripts to validate responses.

# Script Execution Order

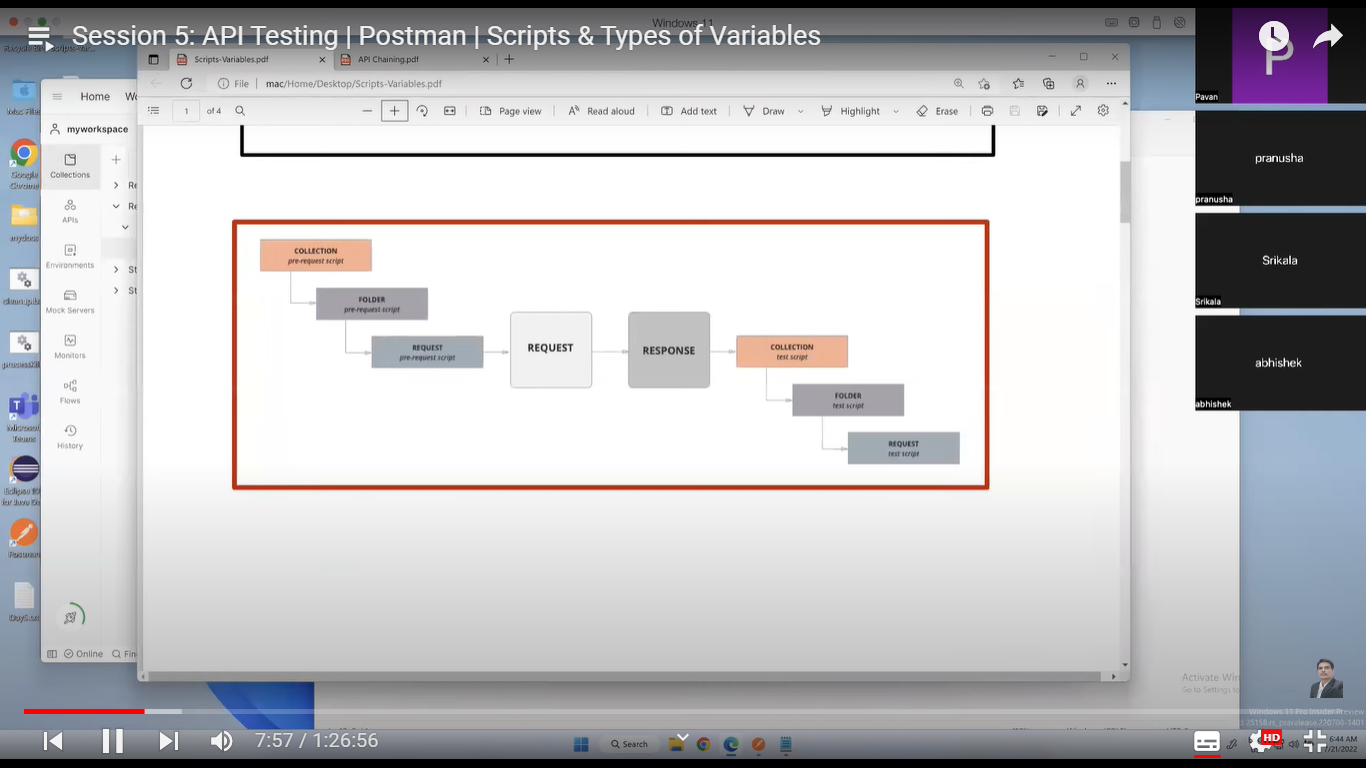
When multiple scripts are present, the execution order follows a hierarchy:

1. **Collection-level scripts**
2. **Folder-level scripts**
3. **Request-level scripts**

This means that collection-level scripts run first, followed by folder-level scripts, and finally request-level scripts.

**Example:** Imagine a collection of API requests to manage users.

* **Collection-level script:** Sets a global authentication token.
* **Folder-level script:** Sets environment variables for a specific user management API.
* **Request-level script:** Validates the response for a particular user creation request.



# Variables in Postman (What / Why / Where)

**Variables** in Postman are dynamic placeholders that store values which can be reused throughout your requests, scripts, and environments. They significantly enhance the reusability and maintainability of your API tests.

# Scope of Variables

The scope determines where a variable can be accessed within Postman.

## Global

* Accessible from anywhere within the workspace.
* Useful for storing values that are constant across all projects.
* Example: API keys, base URLs.

## Collection

* Accessible within a specific collection.
* Suitable for variables shared among requests within a collection.
* Example: Authentication tokens specific to a particular API.

## Environment

* Accessible within all collections but requires switching to the specific environment.
* Ideal for managing different configurations (e.g., development, staging, production).
* Example: Base URLs, API keys for different environments.

## Local

* Accessible only within the pre-request or test script of a specific request.
* Useful for temporary values or calculations.
* Example: Extracting dynamic values from a response to be used in subsequent requests.

## Data

* External files containing data that can be iterated over in your scripts.
* Ideal for data-driven testing.
* Example: CSV files with test data.

**Local Variable Example**

pm.variables.set("myVariable", "value");

This code sets a local variable named myVariable with the value "value".

## Using Variables

To use a variable in a request or script, enclose its name within double curly braces: {{variableName}}.

**Example:** https://api.example.com/users/{{userId}}

# Dynamically Managing Variables in Postman Scripts

## Setting Variables

Postman allows you to dynamically set global, collection, and environment variables directly within your pre-request scripts using the following functions:

* **pm.globals.set("key", "value"):** Creates or updates a global variable with the specified key and value.
* **pm.environment.set("key", "value"):** Creates or updates an environment variable with the specified key and value.
* **pm.collectionVariables.set("key", "value"):** Creates or updates a collection variable with the specified key and value.

## Unsetting Variables

Similarly, you can remove global, collection, and environment variables during the test phase of your request using these functions:

* **pm.globals.unset("key"):** Deletes a global variable with the specified key.
* **pm.environment.unset("key"):** Deletes an environment variable with the specified key.
* **pm.collectionVariables.unset("key"):** Deletes a collection variable with the specified key.

## Accessing Variables in Postman

To access the value of a variable within your Postman scripts, you can use the following functions:

* **pm.globals.get("key"):** Retrieves the value of a global variable.
* **pm.environment.get("key"):** Retrieves the value of an environment variable.
* **pm.collectionVariables.get("key"):** Retrieves the value of a collection variable.

**Example Usage**

// Getting the value of a global variable named 'apiUrl'

**let apiUrl = pm.globals.get("apiUrl");**

// Getting the value of an environment variable named 'token'

**let token = pm.environment.get("token");**

// Getting the value of a collection variable named 'userId'

**let userId = pm.collectionVariables.get("userId");**